

Co-operation in the Parallel Memetic Algorithm

Jakub Nalepa · Mirosław Blocho

Received: 9 September 2014 / Accepted: 23 October 2014 / Published online: 13 November 2014
© The Author(s) 2014. This article is published with open access at Springerlink.com

Abstract Evolutionary algorithms (EAs) have been attracting research attention for last decades. They were shown to be very efficient in solving various complex optimization problems in most fields of science and engineering. In EAs, the population of solutions evolves in time to explore the search space. Parallel EAs became an important stream of development due to a wide availability of parallel computer architectures. Thus, designing parallel algorithms utilizing hundreds of CPU cores efficiently is critical nowadays. In this paper, we investigate the impact of selecting a co-operation scheme for the parallel memetic algorithm (PMA-VRPTW) to solve the NP-hard vehicle routing problem with time windows. In the island-model PMA-VRPTW, which is a hybrid of a genetic algorithm applied to explore the search space, and some refinement methods to exploit solutions already found, a number of populations are evolved in parallel. Processes then co-operate and exchange solutions according to the co-operation scheme (migration policy, interval, and topology). Extensive experimental study (which comprised more than 1,584,000 CPU hours on an SMP cluster) performed on 1000-customer Gehring and Homberger's (GH) benchmark tests gave a detailed insight into the PMA-VRPTW performance and search capabilities. We report 19 (32 % of all 1000-customer GH tests) new world's best solutions obtained using the best co-operation schemes. Finally, we give clear and consistent guidelines on how to select a proper co-operation scheme in PMA-VRPTW based on the test characteristics.

J. Nalepa (✉)

Institute of Informatics, Silesian University of Technology, Akademicka 16,
44-100 Gliwice, Poland
e-mail: jakub.nalepa@polsl.pl

M. Blocho

ABB ISDC, Zeganska 1, 04-713 Warsaw, Poland
e-mail: miroslaw.blocho@pl.abb.com

Keywords Parallel memetic algorithm · Island model · Co-operation · Migration topology · Vehicle routing problem with time windows

1 Introduction

Route scheduling is one of the most important real-life problems and plays a pivotal role in transportation, supply chain management and logistics. Its practical applications encompass the bus route planning, post, food and beverage delivery, cash delivery to banks and ATM terminals, industrial waste collection, maintenance operations, and many more. While constructing the routing schedule for a given distribution problem, it is necessary to consider a large number of practical issues, e.g., the available fleet size, truck capacities, travel costs between geographically dispersed customers, possible time intervals in which customers should be visited, and numerous other circumstances.

Minimizing the number of trucks and their total distance traveled during the service contributes to reducing the fleet exploitation costs and fuel consumption. Also, it lessens the price of delivered goods, since transportation expenses constitute a significant percentage of their value [26]. Moreover, it can help reduce the environmental pollution and traffic congestion, which is an important concern nowadays [29]. Numerous variants of *vehicle routing problems* (VRPs) emerged in order to reflect real-life scheduling scenarios [20]. In the *multiple traveling salesperson problem* (mTSP), which is an extension of a standard *traveling salesperson problem*, more than one salesperson is allowed to be exploited in the solution [7]. Here, each customer has assigned a non-negative demand, which should be satisfied by the visiting salesperson. The fleet size (corresponding to the number of salesmen in mTSP) becomes an important criterion of optimization, apart from the total travel costs. Taking into account the limited capacities of the vehicles led to formulating the *capacitated vehicle routing problem* (CVRP). In CVRP, the fleet is composed of a number of vehicles with well-defined, possibly different load capacities, which cannot be exceeded in a feasible solution. In real-life transportation problems, it is common that customers want to get their orders within a specified time interval. The *vehicle routing problem with time windows* (VRPTW) addresses this issue by incorporating additional constraints concerning delivery time [33].

State-of-the-art techniques to solve the VRPTW are divided into exact and approximate methods. Since the VRPTW is NP-hard, the former approaches can be applied only for relatively small problem instances. Hence, various heuristic algorithms (both sequential and parallel), that do not guarantee obtaining the optimal solution, have been introduced over the years for solving the VRPTW in a reasonable time. They include, among others, simulated annealing [69], tabu search algorithms [28], ant colony systems [26], swarm optimization algorithms [30], evolutionary approaches [57], and many more. Recently, a number of efficient genetic and memetic algorithms—both sequential and parallel—have been proposed for the VRPTW [25, 46, 50, 66].

Memetic algorithms (MAs), which are built upon a population-based approach, and balance exploration with exploitation of the search space, have attracted research attention, and have been applied for solving the VRPTW recently [46, 50, 66]. They

were shown to be very efficient in solving large-scale VRPTW instances. In order to speed up the computations, and to explore larger regions of the search space, we had proposed and recently improved an island-model parallel memetic algorithm for the VRPTW [8, 12, 47–49]. Although we demonstrated its efficacy, its main drawback lies in selecting a proper co-operation scheme of parallel processes (i.e., migration topology, interval, the immigration/emigration policy, and the number of emigrants). In this paper, we carefully investigate the proposed co-operation schemes to determine their impact on the parallel MA performance. Our intensive experimental study conducted on the standard benchmark Gehring and Homberger's (GH) set (with 1000 customers) gave detailed insights into the schemes' performance and behavior. We report 19 new world's best solutions to the GH tests obtained with the use of best co-operation schemes in our parallel MA.

1.1 Contribution

Although parallel EAs (PEAs) proved to be very efficient in solving a variety of optimization problems in many fields, establishing a proper co-operation scheme of parallel processes which ensure good convergence of the algorithm is not a trivial task. Improperly determined migration topology and interval can easily jeopardize the algorithm performance and computation time. In this paper, we propose to apply co-operation schemes which can help improve both exploration and exploitation capabilities of our parallel memetic algorithm (PMA-VRPTW) to solve the largest-scale VRPTW benchmark problem tests (with 1000 customers to serve). We expand our previous research [47], in which we determined three most promising migration topologies for PMA-VRPTW (based on results obtained for 400-customer tests). Here, we apply these schemes in PMA-VRPTW, and perform the in-depth analysis to find out how they influence the algorithm behavior. Also, we incorporate various migration intervals, and investigate their impact on the PMA-VRPTW convergence. We report new world's best solutions for 19 (out of 60) 1000-customer Gehring and Homberger's tests obtained in our extensive experimental study. Finally, we give clear guidelines on how to select the best co-operation scheme in PMA-VRPTW based on the test characteristics.

1.2 Paper Outline

The outline of this paper is as follows. The problem is formally defined in Sect. 2. Section 3 reviews the state-of-the-art algorithms for the VRPTW, and shows current advances in PEAs. Section 4 discusses in detail PMA-VRPTW, along with the co-operation schemes. The results of an extensive experimental study performed on the standard Gehring and Homberger's benchmark set are reported and analyzed in Sect. 5. Finally, Sect. 6 concludes the paper, summarizes the findings, and highlights directions of our future work.

2 Problem Formulation

The VRPTW is a problem of serving M customers by K vehicles of a constant capacity Q . There is a single depot (v_0), which is the start and the finish point of each route. The customers v_i , $i \in \{1, 2, \dots, M\}$, have assigned their own service times s_i , $i \in \{1, 2, \dots, M\}$. Serving the depot does not take any time ($s_0 = 0$), whereas the customer service times are non-negative. A non-negative demand d_i , $i \in \{1, 2, \dots, M\}$, is given for each customer. A geographic dispersion of customers is known, and the travel costs between each pair of travel points are given as $c_{(i,j)}$, where $i \neq j$, and $i, j \in \{0, 1, \dots, M\}$. Additionally, each customer and the depot specifies its earliest and latest time of starting the service (i.e., *time window*), e_i and l_i respectively ($i \in \{0, 1, \dots, M\}$).

Formally, the VRPTW is defined on a directed graph $G = (V, E)$ with a set V of $M + 1$ vertices representing the customers and the depot, along with edges $E = \{\langle v_i, v_{(i+1)} \rangle | v_i, v_{(i+1)} \in V, v_i \neq v_{(i+1)}\}$, representing the travel connections. The i th route is defined as an ordered list of m_i customers served by a single vehicle: $r_i = \langle v_0, v_{(r_i(1))}, \dots, v_{(m_i+1)} \rangle$, where $v_0 = v_{(m_i+1)}$ is the depot, and $v_{(r_i(j))}$ is the j th customer visited within r_i .

2.1 Objectives and Constraints

The primary objective is to minimize K ($K \geq \lceil D/Q \rceil$, where $D = \sum_{i=1}^M d_i$). Secondly, the traveled distance T is to be minimized, where T is given as

$$T = \sum_{i=0}^M \sum_{j=0}^M \sum_{k=1}^K x_{(i,j,k)} c_{(i,j)}. \quad (1)$$

If the k th vehicle travels from v_i to v_j , then $x_{(i,j,k)} = 1$ (0 otherwise).

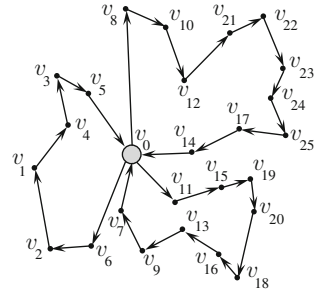
Every customer is visited exactly once, and each route starts and finishes at v_0 . The capacity and the time window constraints must hold for each route. The total amount of goods delivered to customers cannot therefore exceed Q , and the service of each customer must start before its time window closes. If a vehicle visits $v_{(i+1)}$ within its time window ($e_{(i+1)} \leq a_{(i+1)} \leq l_{(i+1)}$, where $a_{(i+1)}$ is the arrival time at $v_{(i+1)}$), then the service is immediate. The vehicle may arrive at $v_{(i+1)}$ before $e_{(i+1)}$, but the service cannot initiate before the time window opens (there is the waiting time $w_{(i+1)}$). If a vehicle visits $v_{(i+1)}$ after closing its time window ($a_{(i+1)} > l_{(i+1)}$), then the service is not feasible.

Let σ_A and σ_B be two VRPTW solutions. Then, σ_A is of a higher quality than σ_B , if $(K(\sigma_A) < K(\sigma_B))$, or $(K(\sigma_A) = K(\sigma_B) \text{ and } T(\sigma_A) < T(\sigma_B))$.

2.2 Example

An exemplary solution σ of the VRPTW instance containing 25 customers is presented in Fig. 1. This solution consists of three routes (r_1 , r_2 , and r_3):

Fig. 1 An exemplary solution to the VRPTW instance with 25 clients served in three routes



$r_1 = \langle v_0, v_8, v_{10}, v_{12}, v_{21}, v_{22}, v_{23}, v_{24}, v_{25}, v_{17}, v_{14}, v_0 \rangle$ (10 customers are visited), $r_2 = \langle v_0, v_{11}, v_{15}, v_{19}, v_{20}, v_{18}, v_{16}, v_{13}, v_9, v_7, v_0 \rangle$ (9 customers), and $r_3 = \langle v_0, v_6, v_2, v_1, v_4, v_3, v_5, v_0 \rangle$ (6 customers). It is easy to see that each customer v_i , $i \in \{1, \dots, 25\}$, is served exactly once. Assuming that the vehicle loads do not exceed the maximum capacity in any route, and the time window constraints are not violated, this routing schedule σ is feasible.

3 Related Literature

3.1 Vehicle Routing Problem with Time Windows

Due to its wide practical applicability, the VRPTW attracted research attention over the years. Although the optimal solutions can be obtained using exact algorithms, their computation time is not acceptable for large-scale problem instances. Thus, a plethora of heuristic algorithms have been proposed, which find high-quality (but not necessarily optimal) solutions in a reasonable time.

Exact algorithms for the VRPTW inherit from works devoted to solving the TSP [33]. In a majority of them, minimizing the travel distance is considered as the single objective. Desrochers et al. proposed to formulate the VRPTW as a set partitioning problem [21], where all feasible routes are considered implicitly. Since the number of possible routes rapidly grows for an increasing number of customers, only a subset of all routes is included in the model. Then, a relaxed shortest path problem is solved to verify if there are feasible routes which decrease the total travel distance. A branch-and-cut procedure for minimizing the number of vehicles was presented in [6]. Irnich and Villeneuve adopted the elementary shortest path problem with resource constraints for the VRPTW [31]. Various VRPTW formulations, including the path, arc, arc-node, and spanning tree formulations, have been studied by several authors [3, 15, 23, 36, 37, 58]. Exact methods were summarized in a bunch of thorough surveys and reviews [4, 19, 22, 33].

Although exact algorithms for solving the VRPTW are still being developed, they are not applicable for large-scale problem instances. Also, they are strongly dependent on the time window characteristics of a specific test case [66]. In contrary to exact methods, in heuristic techniques two VRPTW objectives are usually considered independently. Two-stage techniques (both sequential and parallel), in which K

is minimized at first, and then T is optimized, are of a high research interest. They enable designing effective algorithms for both optimization stages independently.

Heuristic methods can be grouped into construction and improvement techniques. In the former case, unserved customers are iteratively inserted into a partial solution [53–55, 62, 64]. Alternatively, improvement heuristics modify an initial solution to explore new regions of the solution space, e.g., by applying some local search procedures [13, 45]. Meta-heuristic algorithms, which often embed methods for exploring the search space coupled with local search improvement techniques used for its exploitation, allow for existing infeasible solutions and deteriorating their quality temporarily. These approaches include simulated annealing [16], tabu searches [28], ant colony optimization [18], swarm optimization algorithms [30], and many other techniques [5, 13].

EAs have been extensively explored for the VRPTW [57]. In genetic algorithms (GAs), solutions (chromosomes) are successively optimized in the biologically-inspired manner. Chromosomes are selected, crossed-over, and mutated. Memetic algorithms (MAs)—also known as hybrid genetic algorithms—are built upon a similar approach. They combine EAs for the search space exploration, along with refinement procedures applied to exploit solutions already found [46, 50]. Vidal et al. recently proposed an efficient hybrid genetic algorithm which evolves both feasible and infeasible solutions [66]. It is worth mentioning that MAs have been applied to a wide spectrum of other optimization and pattern recognition problems [27, 32, 38, 39, 41, 51].

3.2 Parallel Evolutionary Algorithms

Parallel evolutionary algorithms (PEAs) became a vital research topic thanks to their applicability and availability of various parallel architectures and services, ranging from multicore personal computers, massively parallel graphics processing units, to clusters and computation clouds. The early days of parallel EAs have been summarized in thorough surveys and reviews [2, 14]. The implementation issues of PEAs along with their computation models were discussed in an excellent survey by Alba and Tomassini [1].

There exist a number of parallel computation models defining how distributed algorithm components co-operate during the PEA execution [63]. In the simplest PEAs, parallel algorithm components run independently without any communication. Finally, the results obtained during the execution are collected to determine the highest-quality solution. This model can be considered as a *batch* model, in which sequential executions of an EA are batched and executed on a parallel machine to speed up the computation. In *master-slave* models, a single machine in the system is promoted to act as the master, which distributes the workload to other working machines.

Communication of parallel processes is crucial to guide the search during the PEA execution. Since some processes may have already reached the promising regions of the search space, it is beneficial to distribute this knowledge among other processes which could have stuck in local minima. In the most popular parallel models—*island models* (also referred to as *distributed EAs*)—each process (an *island*) optimizes its population independently from other islands. If the islands run the same EA, then the

system is homogeneous (it is heterogeneous otherwise). The solutions already found in the system are periodically exchanged between the islands. This co-operation is defined by the *migration topology* and *interval* (what is the communication path and when the co-operation is performed), *immigration/emigration policy* (what happens with the obtained/sent solutions), and the number of solutions (migrants) sent between each two islands. The choice of each design setting from the above-mentioned ones is not trivial and significantly affects the PEA performance and behavior. There exist variants of an island model, in which every island contains a single individual in the population, and the mating of an individual is limited to its neighbors. These models, exposing a fine-grain parallelism, are referred to as *cellular EAs*. In co-operative co-evolutionary GAs, optimization problems are divided into subproblems which are independently solved by different parallel populations evolved in (possibly different) EAs. The parallel components communicate later to build a complete solution from concurrently obtained solutions of the subproblems. An excellent review of parallel models and their advantages and disadvantages has been published recently [63].

Similarly to their sequential counterparts, PEAs have been continuously being employed to a plethora of (single- and multi-objective) optimization problems. These applications include, but are not limited to, various scheduling [42,67], routing [68], and assignment problems [40,61], tomography reconstruction [17], classification problems [60], and many others [52,59,65]. Recently, we had proposed and then improved the parallel memetic algorithm for solving the VRPTW [9–12,47–50]

4 Parallel Memetic Algorithm

4.1 Algorithm Outline

In PMA-VRPTW (Algorithm 1), each individual p_i , $i \in \{1, 2, \dots, N\}$, corresponds to a solution σ_i with K routes ($K(p_i) = K(\sigma_i) = K$, and $T(p_i) = T(\sigma_i)$)¹ in a population of size N (on an island). The initial population is generated using a parallel guided ejection search (P-GES). Its sequential counterpart was proposed in [45], and later improved and parallelized in our recent works [49]. P-GES minimizes K at first, and then is used to create initial populations (lines 1–2). They are evolved in PMA-VRPTW to optimize T (lines 3–28).

Following the taxonomy discussed in Sect. 3.2, PMA-VRPTW is classified as a homogeneous island-model parallel memetic algorithm, since each island (a parallel process) runs the same MA to minimize T , and the islands communicate to exchange the knowledge acquired up to date.

4.2 Initial Population

In P-GES (an island-model heuristics), each customer is served within a separate route at first ($K = M$). P-GES repeatedly attempts to decrease K by one at a time.

¹ We will use p_i and σ_i interchangeably in this paper.

Algorithm 1 Parallel memetic algorithm (PMA-VRPTW).

```

1: Minimize the fleet size  $K$  using P-GES;
2: Generate  $N$  solutions with  $K$  routes for each  $P_i, i = 1, \dots, n$  (using P-GES);
3: parfor  $P_i \leftarrow P_1$  to  $P_n$  do
4:    $finished \leftarrow \text{false}$ ;
5:    $z \leftarrow 0$ ; ▷ Initialize the co-operation counter
6:   while not  $finished$  do
7:     Determine  $N$  pairs  $(p_a, p_b)$ ;
8:     for all  $(p_a, p_b)$  do
9:        $p_c^B \leftarrow p_a$ ;
10:      for  $i \leftarrow 1$  to  $N_c$  do
11:         $p_c \leftarrow \text{GenerateChild}(p_a, p_b)$ ; ▷ Fig. 2
12:        if  $T(p_c) < T(p_c^B)$  then
13:           $p_c^B \leftarrow p_c$ ;
14:        end if
15:      end for
16:    end for
17:    Form the next population of size  $N$ ;
18:    if  $z \bmod \delta = 0$  then
19:      Determine emigrant(s) to be sent;
20:      Send selected emigrant(s) to the neighboring island;
21:      Receive immigrant(s) from the neighboring island;
22:      Handle immigrant(s) according to the immigration policy; } Co-operation
23:    end if
24:     $z \leftarrow z + 1$ ; ▷ Increase the co-operation counter
25:    Verify termination condition and update  $finished$ ;
26:  end while
27: end parfor
28: return best solution among all islands;

```

The customers from a random route r are inserted into the *ejection pool* (EP) (a set of unserved customers). The attempts of re-inserting them into the solution are undertaken. If there are no feasible insertions for a customer popped from the EP, then the other ones are removed from σ and put into the EP. The n islands co-operate to exchange their best solutions.

P-GES executes until $K = \lceil D/Q \rceil$ (see Sect. 2), or its computation time exceeds the limit τ_K . Then, it is executed until N solutions with K routes are found for each (out of n) island, or its execution time surpasses τ_N . Finally, the maximum computation time of minimizing K and generating the initial population is $\tau_I = \tau_K + \tau_N$. It is easy to note that PMA-VRPTW is *not* dependent on P-GES, and it can be thus conveniently replaced by another, perhaps more efficient heuristics (either sequential or parallel) to minimize K .

4.3 Selection

Once the initial population is created, it evolves with time to minimize the total travel distance. First, N pairs (p_a, p_b) of individuals from the i th generation G_i are selected to create the $(i+1)$ th generation $G_{(i+1)}$, according to the selection scheme (Algorithm 1, line 7, see also Fig. 2). In this paper, we utilize the AB-selection (AB), which proved to have high exploration capabilities [34, 50]. Here, each individual $p_i, i \in \{1, 2, \dots, N\}$,

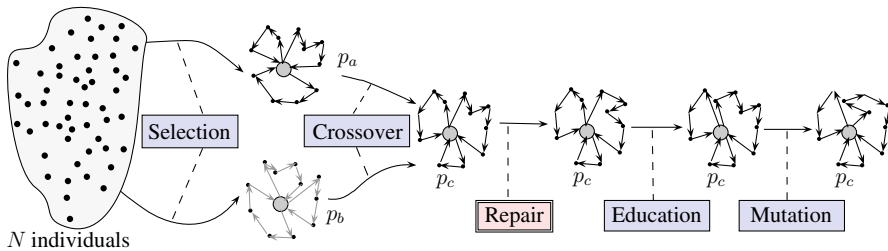


Fig. 2 Creation of a child in each island in PMA-VRPTW

is selected as p_a at first. Then, the individual p'_i is chosen as p_b , such that $p_i \neq p'_i$. Each individual can be selected once as p_a , and once as p_b . This selection takes $O(N)$ time.

4.4 Crossover

For each pair (p_a, p_b) , N_c children p_c are generated using the edge assembly crossover operator (EAX) (Algorithm 1, lines 10–15), firstly introduced for the TSP [43], and later adapted to both the CVRP [44], and the VRPTW [46]. It takes $T_{\text{EAX}}(M) = O(M^2)$ time, where M is a number of customers [8].

The EAX combines two feasible VRPTW solutions σ_a and σ_b (individuals p_a and p_b) consisting of K routes. Let the graphs G_a and G_b correspond to σ_a and σ_b (Fig. 3a, b). Then, the EAX operation comprises the following steps:

1. A new set $((E_a \cup E_b) \setminus (E_a \cap E_b))$, where E_a and E_b are the sets of edges in G_a and G_b (Fig. 3a, b), is found and forms the graph G_{ab} (Fig. 3c).
2. All G_{ab} edges are divided into AB-cycles (Fig. 3d–f), which consist of the G_{ab} edges traced alternately—the E_a edges are traced in the forward direction, whereas the others are traced reversely.
3. The E-set (E_S)—a random AB-cycle (*single mode*, with the probability $\mathcal{P}_s = 0.5$), or a combination of AB-cycles sharing at least one node (*block mode*, $\mathcal{P}_b = 1 - \mathcal{P}_s$)—is found. We took the 2^{nd} AB-cycle (Fig. 3d) as E_S .
4. The intermediate solution is constructed from p_a by removing $E_a \cap E_S$, and adding $E_b \cap E_S$. Then, there are K routes, and possibly some infeasible subroutes, i.e., routes without the depot (Fig. 3g).
5. If the subroutes exist, then a random one to delete is chosen, and its customers are merged with other routes using 2-opt* moves (see Sect. 4.5). This process continues until all subroutes are removed (Fig. 3h).

4.5 Repair, Education, and Mutation

The repair, education, and mutation procedures are the hill-climbing methods based on traditional neighborhoods for VRPTW [35, 46, 56], summarized in Table 1. Here, $v_{(i-1)}$ denotes the predecessor of v_i , and $v_{(i+1)}$ is its successor in the route r_α . Similarly, $v_{(j-1)}$, v_j , and $v_{(j+1)}$ are defined for r_β .

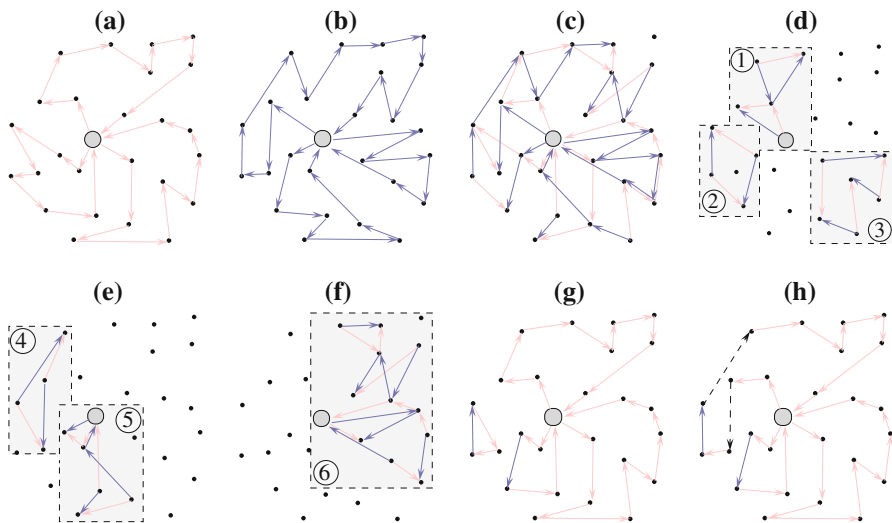


Fig. 3 Illustration of the EAX operator applied to p_a and p_b : **a** the graph G_a corresponding to σ_a (light red), **b** G_b corresponding to σ_b (dark blue), **c** the edges from $E_a \cap E_b$ are removed from $E_a \cup E_b$ to form G_{ab} $((E_a \cup E_b) \setminus (E_a \cap E_b))$, **d–f** six AB-cycles, **g** the intermediate solution $((E_a \setminus (E_a \cap E_S)) \cup (E_b \cap E_S))$, **h** G_c corresponding to p_c (Color figure online)

Table 1 Local search moves applied in PMA-VRPTW

Move	Removed edges	Added edges
2-opt*	$\langle v_i, v_{(i+1)} \rangle, \langle v_j, v_{(j+1)} \rangle$	$\langle v_i, v_{(j+1)} \rangle, \langle v_j, v_{(i+1)} \rangle$
Out-exchange	$\langle v_{(i-1)}, v_i \rangle, \langle v_i, v_{(i+1)} \rangle,$ $\langle v_{(j-1)}, v_j \rangle, \langle v_j, v_{(j+1)} \rangle$	$\langle v_{(i-1)}, v_j \rangle, \langle v_j, v_{(i+1)} \rangle,$ $\langle v_{(j-1)}, v_i \rangle, \langle v_i, v_{(j+1)} \rangle$
Out-relocate	$\langle v_{(i-1)}, v_i \rangle, \langle v_i, v_{(i+1)} \rangle,$ $\langle v_{(j-1)}, v_j \rangle$	$\langle v_{(j-1)}, v_i \rangle, \langle v_i, v_j \rangle,$ $\langle v_{(i-1)}, v_{(i+1)} \rangle$
In-exchange	$\langle v_{(i-2)}, v_{(i-1)} \rangle, \langle v_{(i-1)}, v_i \rangle,$ $\langle v_i, v_{(i+1)} \rangle, \langle v_{(i+1)}, v_{(i+2)} \rangle$	$\langle v_{(i-2)}, v_{(i+1)} \rangle, \langle v_{(i+1)}, v_i \rangle,$ $\langle v_i, v_{(i-1)} \rangle, \langle v_{(i-1)}, v_{(i+2)} \rangle$
In-relocate	$\langle v_{(i-1)}, v_i \rangle, \langle v_i, v_{(i+1)} \rangle,$ $\langle v_{(i+1)}, v_{(i+2)} \rangle$	$\langle v_{(i-1)}, v_{(i+1)} \rangle, \langle v_{(i+1)}, v_i \rangle,$ $\langle v_i, v_{(i+2)} \rangle$
GENIUS-exchange	$\langle v_{(i-1)}, v_i \rangle, \langle v_i, v_{(i+1)} \rangle,$ $\langle v_{(j-2)}, v_{(j-1)} \rangle, \langle v_j, v_{(j+1)} \rangle,$ $\langle v_{(j+1)}, v_{(j+2)} \rangle$	$\langle v_{(i-1)}, v_{(i+1)} \rangle, \langle v_{(j-2)}, v_i \rangle,$ $\langle v_i, v_{(j+1)} \rangle, \langle v_{(j+1)}, v_{(j-1)} \rangle,$ $\langle v_j, v_{(j+2)} \rangle$

Let $\mathcal{N}(\sigma)$ be the neighborhood of σ obtained by applying the mentioned moves. Since it encompasses a huge number of solutions, we limit its size by considering N_{v_i} nearest customers for v_i in each move. Calculating the load change after a move is obvious ($O(1)$ time), whereas to evaluate the time window penalty we use the approach proposed in [46] ($O(1)$ for 2-opt*, out-exchange, and out-relocate, $O(m)$ for other moves, where m is the route size).

If σ_c is infeasible, then it needs to be repaired (double-lined box in Fig. 2), by performing local search moves to decrease the penalty $\xi(\sigma) = \beta_1 P_c(\sigma) + \beta_2 P_{tw}(\sigma)$, where $P_c(\sigma)$ and $P_{tw}(\sigma)$ represent the capacity and time window violations, multiplied by some coefficients β_1 and β_2 . We set $\beta_1 = \beta_2 = 1.0$, and $N_{vi} = 50$ as suggested in [46]. An infeasible route r is selected randomly from σ_c at first. Then, the set $\bigcup_{v \in r} \mathcal{N}(\sigma_c, v)$ of subneighborhoods created for each customer that violates the constraints within r , is determined. Finally, a new solution $\sigma'_c, \sigma'_c \in \bigcup_{v \in r} \mathcal{N}(\sigma_c, v)$, which minimizes $\xi(\sigma)$, replaces σ_c . This process executes until σ_c is feasible, or there are no further repair moves.

If p_c is feasible, then it is educated by feasible moves decreasing $T(p_c)$. If there are no more improvement moves, the process stops. p_c is then mutated by at most I_M feasible moves (in this work we set $I_M = 300$). The best feasible child p_c^B for this pair of parents is updated if it is necessary (Algorithm 1, line 13). Finally, p_c^B replaces p_a in the next generation if $T(p_c^B) < T(p_a)$ (line 17).

The crossover, repair and education procedures—being the most time-consuming parts of the MA run by each island in PMA-VRPTW—require $O(M^2)$ time. Performing these operations for N pairs of parents to generate N_c children takes $\mathcal{T}_G = c_o N M^2$ for a constant c_o . The maximum number of generations is bounded by another constant c_g , $c_o < c_g$, due to the execution time limit. Hence, the time complexity of the MA run by each island is $O(M^2)$.

It is worth to mention that we do not incorporate any diversity management procedures into the MA run by each island (e.g., regenerating the population if it encounters the *diversity crisis*). We avoid introducing new genetic material during the PMA-VRPTW execution in order to better verify how it copes with the local minima faced during the optimization (especially while applying various co-operation schemes discussed in Sect. 4.6).

4.6 Co-operation

The n islands in PMA-VRPTW co-operate periodically (according to the migration interval δ , Algorithm 1, line 18) to guide the search towards solutions of a better quality. This process is highlighted by a curly brace in Algorithm 1 (lines 19–22). First, the emigrants (i.e., solutions to be sent from the island) are determined (line 19). The emigrants are then transferred to the neighboring island (according to the migration topology) (line 20). This operation may be done either synchronously or asynchronously. In the former case, the sending island waits until the send operation is finished. In this paper, we send the emigrants asynchronously, i.e., the algorithm execution at the sending island proceeds while the communication progresses. Additionally, the selected emigrants are transferred only if they have been updated since the previous co-operation (to avoid unnecessary data transfer).

After acquiring (one or more) immigrants, they are handled by the receiving island according to the policy defined by the co-operation scheme (line 22). They may (i) be appended to the population, (ii) replace some individuals at the receiving island, or (iii) be crossed-over with other individuals.

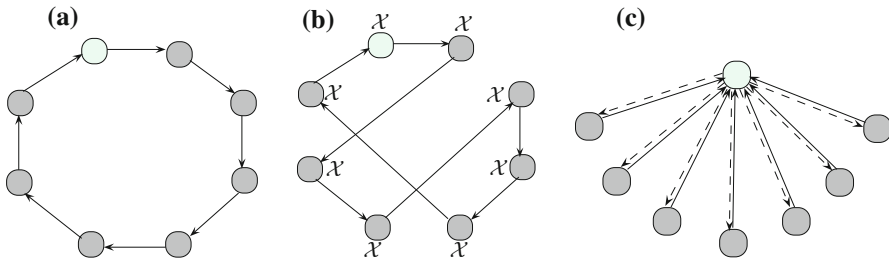


Fig. 4 Migration topologies: **a** Ring, **b** R-EAX (\mathcal{X} denotes the EAX operation), **c** KS. The master process (island) is rendered in *light green* (Color figure online)

In our earlier work [47], we investigated different co-operation schemes:

1. **Independent runs.** Each island runs the MA independently without any co-operation. Finally, solutions are compared and the best one is chosen.
2. **Pool.** Each island sends to the nominated master island the best p^B in its population, and it is inserted into the pool. The master determines ηN , $0 < \eta < 1$, best solutions, which replace ηN random ones in each island.
3. **Pool with EAX (P-EAX).** The regular pool is enhanced by the EAX operator employed while handling ηN immigrants by each island. The solution p_a —that is being replaced—is crossed-over with the immigrant p^I , if $p_a \neq p^I$, to generate p_c^I . If p_c^I is feasible, then it replaces p_a .
4. **Ring.** The migration topology constitutes a ring.
5. **Randomized EAX (R-EAX).** The regular ring is enhanced by the EAX operator. Also, the migration topology (order of islands in the ring) is randomized, and determined by the master before each co-operation.
6. **Knowledge synchronization (KS).** The nominated master island synchronizes the knowledge acquired by all islands during the search (i.e., distributes the best solutions among the islands).

An initial experimental study performed on 400-customer GH tests clearly indicated that the three last schemes (i.e., Ring, R-EAX, and KS) significantly outperformed the other ones in terms of the convergence time, and balancing exploration and exploitation capabilities of PMA-VRPTW [47]. The corresponding migration topologies of these schemes are given in Fig. 4, along with the other characteristics highlighted in Table 2.

Although the initial experiments gave a rough overview of the co-operation schemes' performance, we did not consider the impact of the migration interval on the final results. Also, most demanding tests (with 1000 customers) were not investigated. Therefore, in this paper we aim at performing an in-depth analysis of the PMA-VRPTW behavior for each co-operation scheme. Additionally, we define two migration intervals ($\delta = 2$ and $\delta = 20$, given in the number of consecutive generations) to verify their impact on final solutions, and exploration and exploitation capabilities of the parallel MA.

It is easy to see that the time complexity of PMA-VRPTW includes the time complexity of the applied co-operation scheme (see \mathcal{T}_C in Table 2), and is given

Table 2 Summary of the investigated co-operation schemes

Characteristic ↓	Ring	R-EAX	KS
Migration topology	Figure 4a	Figure 4b	Figure 4c
Number of emigrants	1 (p^B)	1 (p^B)	1 (p^B)
Number of immigrants	1	1	$N_b, N > N_b \geq 1$
Immigration policy	Replace p^B , if $T(p^B) > T(p^I)$	Replace p^B , if $T(p^B) > T(p^I)$, otherwise $p_c^I \leftarrow \mathcal{X}(p^B, p^I)$, and p_c^I replaces p^B , if $T(p^B) > T(p_c^I)$	Replace N_b worst individuals
Emigration policy	Keep emigrants in the sending island's population		
Migration interval (δ)	$\delta = 2$ or $\delta = 20$ (the number of consecutive generations)		
Time complexity (\mathcal{T}_C) ^a	nM	nM^2	$2nM + n \log n$

^a Time complexity of a single co-operation phase

as $\mathcal{T}_{\text{PMA}}(n, M) = c_p(c_o n M^2 + c_c \mathcal{T}_C)$, where c_p and c_c are some constants. Since we impose the time limit on PMA-VRPTW (τ_P), c_p is bounded by another constant (see the analysis in Sect. 4.5). Thus—in the worst case—the time complexity of PMA-VRPTW is $O(M^2 + nM^2)$ (for a constant population size N), if the R-EAX co-operation scheme is applied (see Table 2).

5 Experimental Results

5.1 Setup

The PMA was implemented in C++ using the Message Passing Interface (MPI) library. The source code was compiled using Intel 10.1 compiler and MVAPICH1 v0.9.9 MPI library. The performance experiments were conducted on Galera supercomputer (<http://task.gda.pl/hpc-en/>), whose total theoretical peak performance has been estimated to 50 TFLOPS. It is equipped with 1344 Intel Xeon Quad Core 2.33 GHz processors (5376 cores), each with 12 MB level 3 cache. The nodes were connected by the Mellanox InfiniBand DDR fat-free interconnect (throughput 20 Gbps, delay 5 μ s). This supercomputer was executing Linux operating system.

PMA-VRPTW was executed on 96 processors² ($n = 96$), and its maximum execution time limits were as follows: $\tau_K = 10$ min., $\tau_N = 60$ min. (thus, $\tau_I = \tau_K + \tau_N = 70$ min.), and $\tau_P = 660$ min. (see Sect. 4 for more details). The population size and the number of children generated for each pair of parents in the MA (run by each island) were not changed (i.e., *controlled*) during the PMA-VRPTW execution, and were experimentally tuned to the following values: $N = 100$, and $N_c = 20$. For each GH instance, the algorithm was run 5 times for each co-operation scheme (and each

² Either a uniprocessor or a core of a multicore processor.

Table 3 Overview of Gehring and Homberger’s test characteristics

Characteristic ↓	C1	C2	R1	R2	RC1	RC2
Structure	Clustered	Clustered	Random	Random	Mixed	Mixed
Time windows	Tight	Wide	Tight	Wide	Tight	Wide
Vehicle capacities	Small	Large	Small	Large	Small	Large

migration interval). It gives at least³ $(96 \text{ processors}) \times (660 \text{ min.}) \times (5 \text{ runs}) \times (50 \text{ GH tests}) \times (3 \text{ co-operation schemes}) \times (2 \text{ migration intervals}) = 1,584,000$ CPU hours of the entire experimental study.

5.2 Dataset

PMA-VRPTW was tested on a classical benchmark set proposed by Gehring and Homberger [24], which reflects various real-life scheduling circumstances. All large-scale tests are split into subclasses, containing customers grouped into clusters (C subclass), dispersed randomly on the map (R subclass), and those with a mix of clustered and randomized customers (RC subclass). There are problems with relatively small vehicle capacities and short time windows (C1, R1, and RC1), and those with larger vehicle capacities and a longer scheduling horizon (C2, R2, and RC2) (see Table 3). These characteristics strongly influence the structure of final solutions, e.g., a larger fleet is necessary to serve customers in the case of small truck capacities and tight time windows.

There are problems with various M ’s, $M \in \{200, 400, 600, 800, 1000\}$. Each Gehring and Homberger’s (GH) subclass contains 10 problems (60 instances in total for each M). Tests are distinguished by their unique names: $\alpha_ \beta_ \gamma$, where α denotes the subclass, β relates to M (2 for 200, 4 for 400, and so forth), and γ is the test identifier ($\gamma \in \{1, 2, \dots, 10\}$). In this study, we focus on the most demanding 1000-customer GH tests. The world’s best (currently known) results for GH tests are summarized at the SINTEF website.⁴ Note that the world’s best results is a set of solutions obtained using *various* algorithms (both sequential and parallel)—see the SINTEF website for details.

5.3 Analysis and Discussion

The first stage of PMA-VRPTW consists in minimizing the number of vehicles (K). We utilized the parallel guided ejection search (P-GES) to optimize K at first (within the time τ_K), and then to generate an initial population of N solutions for each island (within τ_N . For details see Sect. 4). If τ_N appears not enough to generate N solutions

³ We exclude the P-GES execution time here. Also, we consider only 50 (out of 60) GH tests for which PMA-VRPTW was run to optimize T (see Sect. 5.3).

⁴ <http://www.sintef.no/Projectweb/TOP/VRPTW/>; reference date: August 28, 2014.

for an island, then the remaining solutions are obtained by mutating the already-found individuals (i.e., by applying 200 local search moves) (see Sect. 4.5).

It is worth noting, that solutions with lower K are usually characterized by a larger travel distance T (if $K_\alpha < K_\beta$ then $T_\alpha > T_\beta$ in most cases). Since we focus on the travel distance minimization in PMA-VRPTW, we omit 10 GH instances⁵ in the analysis, for which P-GES did not manage to find solutions with $K = K_{WB}$, where K_{WB} is the world's minimum number of routes, in order to provide a fair comparison. For each of the mentioned tests P-GES ended up with solutions containing $K = K_{WB} + 1$ routes (for other tests $K = K_{WB}$). Despite its significant time complexity [8], P-GES runs very fast in practice—the average times of generating a single individual in a population were (given in seconds): $\tau_K^A = 28.73$ (C1 class), $\tau_K^A = 91.74$ (C2), $\tau_K^A = 6.69$ (R1), $\tau_K^A = 17.53$ (R2), $\tau_K^A = 6.40$ (RC1), and $\tau_K^A = 181.80$ (RC2). These times are neglectable compared to the execution time of PMA-VRPTW. As already mentioned, PMA-VRPTW is independent from P-GES, and it can be easily replaced by a more efficient route minimization algorithm.

The best results (out of 5 independent runs) obtained using PMA-VRPTW for each GH instance, along with the best results averaged across the subclasses (for each problem instance) are given in Tables 4 and 5, for $\delta = 2$ and $\delta = 20$, respectively. The best T 's among co-operation schemes are rendered in boldface. Also, we present the results that are better than the current world's best ones (they are indicated by the \star symbol). It is easy to see that the Ring and KS schemes significantly outperformed R-EAX for both migration intervals. This clearly indicates that the additional crossover of solutions (the immigrant and the best individual in an island's population) does not result in a significant improvement of final solutions. However, for clustered customers with tight time windows (C1 class), the EAX structural changes appeared beneficial and R-EAX slightly outranked the other schemes (Ring was outdone by 0.24 % for $\delta = 2$, and 0.05 % for $\delta = 20$, and KS by 0.13 % for $\delta = 2$, and 0.15 % for $\delta = 20$). This, in turn, shows that the local changes (resulting from the education and mutation) of best individuals were not able to improve locally-optimal solutions of tests with geographically grouped customers (and that the best solutions are close to the “feasibility border”). In contrary, both Ring and KS gave much better results than R-EAX for other GH subclasses. It is worth mentioning that PMA-VRPTW converged to the solutions better than already published for the entire R2 subclass, and for the majority of RC2 tests (see Ring and KS in Tables 4 and 5).

The average results (out of 5 runs for each configuration) are given in Tables 6 and 7 (similarly, the best results⁶ are rendered in boldface, and those better than the world's best ones are annotated with \star). The results show that Ring and KS provide the most stable results, and are able to guide PMA-VRPTW to asymptotically similar solutions (R-EAX reached very high-quality C1 solutions only). For frequent co-operation ($\delta = 2$), KS turned out to be the best scheme on average, and managed to compensate

⁵ C1_10_6, C1_10_7, C1_10_8, C2_10_3, C2_10_6, C2_10_7, C2_10_8, RC2_10_1, RC2_10_2, and RC2_10_5.

⁶ Not necessarily statistically significant. In order to perform significance tests, PMA-VRPTW would have to be executed much more number of times (e.g., 100 for each GH instance). Given a very large τ_P , it would take an enormous amount of time, which is far beyond the scope of this work.

Table 4 The best results (out of 5 runs) obtained using PMA-VRPTW with various co-operation schemes ($\delta = 2$)

Id ↓	C1	C2	R1	R2	RC1	RC2
<i>Ring</i>						
1	42,478.95	16,879.24	53,687.82	42,188.86*	46,725.88	–
2	42,387.85	17,126.39	49,630.74	33,634.05	44,494.17	–
3	40,446.33	–	45,793.13	24,965.63*	42,772.49	20,050.71*
4	39,659.12	15,742.92	43,320.69	17,926.45*	41,835.34	15,869.82
5	42,469.18	16,561.57	52,186.14	36,243.51*	45,981.43	–
6	–	–	48,411.62	30,123.53*	45,892.32	26,910.35*
7	–	–	45,050.20	23,257.36*	45,245.68	25,116.15*
8	–	–	43,017.17	17,585.79*	44,565.66	23,845.66
9	41,031.46	16,421.32*	51,042.39	33,064.26	44,841.57	23,112.13*
10	42,478.95	15,982.98	48,986.21	30,257.93*	44,214.03	22,048.98
Avg.	41,251.06	16,452.40	48,112.61	28,924.74	44,656.86	22,421.97
<i>R-EAX</i>						
1	42,478.95	16,879.24	53,772.62	42,261.34	46,612.69	–
2	42,278.45*	17,126.39	49,832.97	33,570.65	44,781.82	–
3	40,338.02	–	46,237.88	25,029.43*	42,830.96	20,132.61
4	39,589.01	15,805.27	43,659.12	19,180.23	42,043.04	15,992.44
5	42,469.18	16,563.10	52,713.43	36,289.00*	45,850.40	–
6	–	–	48,664.85	30,270.10	45,889.93	27,097.58
7	–	–	45,393.80	23,730.70	45,382.16	25,380.63
8	–	–	43,632.43	18,018.54	44,780.84	24,013.56
9	40,733.69	16,421.14*	51,364.94	33,146.76	44,757.77	23,292.42
10	40,167.29	16,022.40	49,411.66	30,428.55*	44,273.97	22,225.09
Avg.	41,150.66	16,469.59	48,468.37	29,192.53	44,720.36	22,590.62
<i>KS</i>						
1	42,478.95	16,879.24	53,676.62	42,196.47*	46,694.35	–
2	42,339.90	17,126.39	49,877.76	33,550.62*	44,371.97	–
3	40,343.51	–	45,749.74	24,989.17*	42,774.24	20,068.74
4	39,548.04	15,829.51	43,226.05	18,262.09	41,873.12	15,790.47
5	42,469.18	16,561.57	52,328.30	36,250.95*	45,962.10	–
6	–	–	48,386.76	30,102.91*	45,622.36	26,883.75*
7	–	–	45,001.07	23,270.90*	45,358.51	25,162.18*
8	–	–	43,101.84	17,526.72*	44,684.21	23,822.03
9	40,898.81	16,372.32*	50,882.21	33,025.59*	44,532.34	23,089.01*
10	40,357.13	15,988.14	49,004.42	30,270.36*	44,287.86	22,004.78*
Avg.	41,205.07	16,459.53	48,123.48	28,944.58	44,616.11	22,402.99

The best result for each instance (among co-operation schemes) is rendered in boldface; $T^* < T_{WB}$ (T_{WB} is the world's best currently known T)

Table 5 The best results (out of 5 runs) obtained using PMA-VRPTW with various co-operation schemes ($\delta = 20$)

Id ↓	C1	C2	R1	R2	RC1	RC2
<i>Ring</i>						
1	42,478.95	16,879.24	53,560.85*	42,188.86*	46,281.52	–
2	42,291.63*	17,126.39	49,350.91	33,554.71*	44,441.19	–
3	40,325.00	–	45,822.97	24,954.81*	42,782.41	20,084.70
4	39,638.56	15,755.22	43,152.26	17,997.69*	41,951.81	15,770.18
5	42,469.18	16,561.57	52,230.01	36,251.43*	45,801.99	–
6	–	–	48,187.98	30,099.45*	45,674.32	26,864.81*
7	–	–	45,099.17	23,344.99*	45,164.73	25,112.77*
8	–	–	43,059.13	17,495.51*	44,679.94	23,709.29*
9	40,709.26	16,370.44*	50,653.74	33,019.95*	44,498.21	23,063.76*
10	40,296.60	15,981.89	48,779.67	30,242.35*	44,227.46	21,965.94*
Avg.	41,172.74	16,445.79	47,989.67	28,914.98	44,550.36	22,367.35
<i>R-EAX</i>						
1	42,478.95	16,879.24	53,846.08	42,259.70	46,833.99	–
2	42,289.09*	17,126.39	50,190.39	33,912.29	44,772.67	–
3	40,340.90	–	46,280.64	25,106.05	42,877.25	20,229.46
4	39,555.09	15,908.75	44,507.48	19,421.25	42,400.06	17,124.27
5	42,469.18	16,563.10	52,659.74	36,352.12	45,927.56	–
6	–	–	48,919.69	30,309.43	45,827.36	27,086.98
7	–	–	45,808.16	23,636.36	45,418.16	25,471.85
8	–	–	43,528.46	17,859.14	44,879.47	24,063.00
9	40,694.49	16,481.81	51,494.07	33,116.32*	44,878.70	23,297.83
10	40,231.23	16,116.81	49,742.52	30,499.52*	44,247.53	22,347.35
Avg.	41,151.28	16,512.68	48,697.72	29,247.22	44,806.28	22,802.96
<i>KS</i>						
1	42,478.95	16,879.24	53,615.67*	42,188.86*	46,496.78	–
2	42,338.01	17,126.39	49,671.22	33,512.83*	44,365.22	–
3	40,310.85	–	45,719.52	24,940.32*	42,790.99	20,076.27
4	39,710.03	15,746.46	43,291.04	17,959.91*	41,780.75	15,754.41
5	42,469.18	16,561.57	52,218.87	36,232.18*	45,910.92	–
6	–	–	48,335.40	30,091.93*	45,722.35	26,797.76*
7	–	–	45,080.30	23,355.92*	45,096.25	25,128.19*
8	–	–	43,156.24	17,524.42*	44,665.94	23,796.20
9	40,910.39	16,372.32	50,768.29	33,002.36*	44,643.41	23,061.90*
10	40,283.92	15,993.24	48,946.49	30,215.24*	44,235.87	21,999.00*
Avg.	41,214.48	16,446.54	48,080.30	28,902.40	44,570.85	22,373.39

The best result for each instance (among co-operation schemes) is rendered in boldface; $T^* < T_{WB}$ (T_{WB} is the world's best currently known T)

Table 6 The average results (out of 5 runs) obtained using PMA-VRPTW with various co-operation schemes ($\delta = 2$)

Id ↓	C1	C2	R1	R2	RC1	RC2
<i>Ring</i>						
1	42,478.95	16,879.24	53,735.17	42,212.86*	46,954.02	–
2	42,387.85	17,126.39	49,726.08	33,670.45	44,744.58	–
3	40,528.58	–	45,941.36	25,079.85	42,928.15	20,160.25
4	39,759.69	16,089.40	43,367.80	18,014.51*	41,931.37	15,930.61
5	42,469.18	16,562.49	52,346.05	36,272.59*	46,029.71	–
6	–	–	48,669.45	30,222.22*	46,076.97	26,912.34*
7	–	–	45,204.57	23,424.77	45,460.11	25,116.15*
8	–	–	43,107.99	17,628.96*	44,894.12	23,881.79
9	41,266.71	16,450.34	51,280.59	33,093.84*	44,897.74	23,146.48*
10	40,438.51	16,023.14	49,125.08	30,296.18*	44,449.37	22,120.70
Avg.	41,332.78	16,521.83	48,250.41	28,991.62	44,836.61	22,466.90
<i>R-EAX</i>						
1	42,478.95	16,879.24	53,807.59	42,291.11	46,961.67	–
2	42,406.28	17,142.98	50,178.31	33,640.11	44,902.27	–
3	40,392.98	–	46,363.70	25,191.62	42,955.13	20,224.91
4	39,654.39	15,868.05	43,809.30	19,433.22	42,097.47	16,106.00
5	42,469.18	16,566.20	52,952.11	36,347.26	46,062.40	–
6	–	–	48,883.58	30,352.69	46,080.32	27,146.39
7	–	–	45,554.95	23,803.27	45,512.01	25,494.36
8	–	–	43,791.72	18,182.23	44,870.17	24,164.79
9	40,912.96	16,498.40	51,876.63	33,202.82	44,873.52	23,438.13
10	40,325.16	16,059.15	49,726.29	30,563.48*	44,382.68	22,427.01
Avg.	41,234.27	16,502.34	48,694.42	29,300.78	44,869.76	22,714.51
<i>KS</i>						
1	42,478.95	16,879.24	53,691.52	42,214.16*	46,795.24	–
2	42,668.32	17,126.39	49,960.81	33,615.75	44,546.19	–
3	40,402.24	–	45,851.78	25,038.87*	42,968.44	20,169.93
4	39,736.93	15,873.11	43,339.93	18,365.29	41,985.52	15,852.43
5	42,469.18	16,561.57	52,602.36	36,261.01*	46,008.60	–
6	–	–	48,625.45	30,171.57*	45,897.57	26,908.14*
7	–	–	45,176.82	23,412.49	45,505.78	25,252.48*
8	–	–	43,199.05	17,582.20*	44,809.70	23,883.43
9	40,970.05	16,400.61*	51,024.38	33,060.30*	44,779.06	23,161.29
10	40,477.13	15,999.76	49,197.66	30,313.90*	44,413.41	22,065.36*
Avg.	41,314.69	16,473.45	48,266.98	29,003.55	44,770.95	22,470.44

The best result for each instance (among co-operation schemes) is rendered in boldface; $T^* < T_{WB}$ (T_{WB} is the world's best currently known T)

Table 7 The average results (out of 5 runs) obtained using PMA-VRPTW with various co-operation schemes ($\delta = 20$)

Id ↓	C1	C2	R1	R2	RC1	RC2
<i>Ring</i>						
1	42,478.95	16,879.24	53,626.82*	42,209.68*	46,562.32	–
2	42,359.36	17,126.39	49,729.14	33,647.83	44,615.58	–
3	40,431.26	–	45,921.21	24,999.15*	42,834.01	20,125.28
4	39,744.58	15,832.46	43,330.69	18,056.59	42,014.82	15,798.70
5	42,469.18	16,562.49	52,359.55	36,255.76*	46,022.34	–
6	–	–	48,387.30	30,178.59*	45,760.65	26,879.59*
7	–	–	45,251.39	23,387.73	45,292.69	25,138.07*
8	–	–	43,146.94	17,546.31*	44,732.74	23,803.10
9	40,984.30	16,379.48*	50,901.63	33,047.26*	44,722.24	23,100.65
10	40,369.25	16,010.32	48,980.19	30,284.52*	44,406.02	21,995.50*
Avg.	41,262.41	16,465.06	48,163.49	28,961.34	44,696.34	22,405.84
<i>R-EAX</i>						
1	42,478.95	16,879.24	53,935.61	42,270.62	47,182.99	–
2	42,367.15	17,126.39	50,459.87	33,951.40	44,820.79	–
3	40,386.08	–	46,426.27	25,170.14	43,049.98	20,363.40
4	39,648.46	15,958.67	44,732.58	19,492.46	42,475.18	17,180.56
5	42,469.18	16,569.10	52,950.27	36,391.82	46,365.72	–
6	–	–	49,096.64	30,413.00	46,144.14	27,131.66
7	–	–	45,931.75	23,789.55	45,591.49	25,471.85
8	–	–	43,731.51	17,908.58	45,014.27	24,199.93
9	40,876.66	16,524.75	51,782.76	33,252.21	44,995.33	23,496.33
10	40,379.99	16,137.12	49,969.36	30,659.73	44,456.08	22,441.98
Avg.	41,229.50	16,532.55	48,901.66	29,329.95	45,009.60	22,897.96
<i>KS</i>						
1	42,478.95	16,879.24	53,649.09*	42,201.83*	46,779.46	–
2	42,422.66	17,126.39	50,092.45	33,654.05	44,611.65	–
3	40,421.20	–	45,856.74	24,980.55*	42,938.35	20,128.92
4	39,819.31	15,826.95	43,616.72	18,292.28	42,021.67	15,992.54
5	42,469.18	16,561.57	52,412.51	36,247.61*	46,085.55	–
6	–	–	48,477.39	30,163.96*	45,920.41	26,839.08*
7	–	–	45,244.39	23,427.09	45,336.30	25,166.34*
8	–	–	43,226.30	17,558.80*	44,781.20	23,837.32
9	40,980.37	16,383.74*	51,044.55	33,043.74*	44,746.04	23,076.81*
10	40,436.61	15,998.07	49,097.84	30,245.51*	44,321.03	22,047.55*
Avg.	41,289.75	16,462.66	48,271.80	28,981.54	44,754.17	22,441.22

The best result for each instance (among co-operation schemes) is rendered in boldface; $T^* < T_{WB}$ (T_{WB} is the world's best currently known T)

Table 8 Summary of the results obtained using PMA-VRPTW with various co-operation schemes: $\#T_B$ —number of the best T 's obtained with the corresponding co-operation scheme, $\%T_B$ —percentage of the best T 's, $\#T^*$ —number of T 's such that $T < T_{WB}$ (the meaning of the results with A superscript—averaged T 's, see Tables 6 and 7—are analogous)

Scheme \rightarrow	Ring		R-EAX		KS	
	2	20	2	20	2	20
$\#T_B$ (out of 50)	24	26	11	8	24	26
$\%T_B$ (in %)	48	52	22	16	48	52
$\#T^*$	14	18	5	3	14	15
$\#T_B^A$ (out of 50)	19	32	9	7	30	19
$\%T_B^A$ (in %)	38	64	18	14	60	38
$\#T^{A*}$	10	12	1	0	11	13

the co-operation overhead by synchronizing the already-gained knowledge across the islands (this scheme is very exploitative). On the other hand, Ring is shown to be an explorative scheme, which is able to converge to the best results with relatively large migration interval (i.e., rare co-operation) by exploring a large part of the search space ($\delta = 20$). A noteworthy observation is that PMA-VRPTW is able to improve the world's best results of wide-time-window tests (R2 and RC2) with a very large probability (70 % of R2, and 43 % of RC2 average results have $T < T_{WB}$, in the case of both Ring and KS).

The results—both average and best—obtained using various co-operation schemes are summarized in Table 8 (we consider only tests for which $K = K_{WB}$). In boldface are indicated the best results across co-operation schemes (separately for $\delta = 2$ and $\delta = 20$). The results confirm that KS is the most appropriate scheme for frequent co-operation, whereas Ring for a rarer one. KS is very stable and converged to high-quality results (see $\#T^{A*}$). In Table 9, we present the best asymptotic results for selected GH instances (i.e., those for which PMA-VRPTW obtained results with $T < T_{WB}$)—the new world's best results are shown in boldface.⁷ Although Ring appeared to be very competitive (see $\#T_B$, $\%T_B$, and $\#T^*$ in Table 8), it is KS (with $\delta = 20$) which gave the asymptotically best solutions (Table 9). Thus, the exploration of the search space is well-balanced with exchanging the best solutions and their further exploitation. Too frequent migration ($\delta = 2$) results in saturating islands with similar solutions which are not further improved during the search.

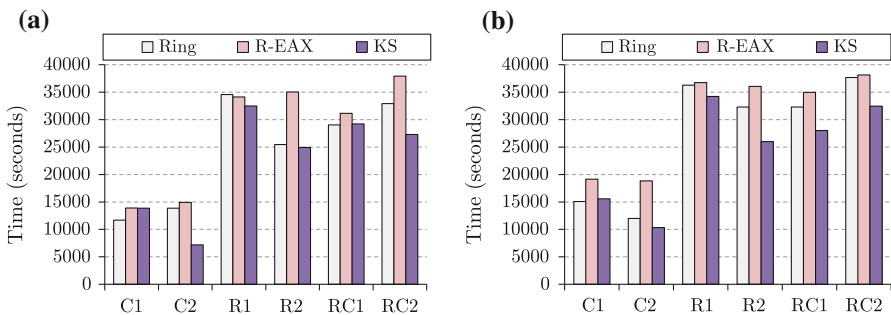
In Fig. 5, we present the average convergence time (i.e., after which the best individual among all islands has not been improved) of PMA-VRPTW. It is easy to see that C1 and C2 tests can be solved much faster compared with the other GH instances. Here, applying R-EAX scheme is beneficial in terms of the solutions quality (it gave best asymptotic results), but it also increases the convergence time of PMA-VRPTW. Since R-EAX requires crossing-over individuals (which may result in infeasible children), and restoring their feasibility if necessary (in a time-consuming repair process),

⁷ See details of these solutions at <http://sun.uci.aci.polsl.pl/~jnalepa/VRPTW>.

Table 9 The best results (out of 5 runs) obtained using PMA-VRPTW with various co-operation schemes for selected GH benchmark tests (for which PMA-VRPTW ended up with a solution such that $T < T_{WB}$)

Scheme →	Ring		R-EAX		KS		WB
Test ↓ $\delta \rightarrow$	2	20	2	20	2	20	
C1_10_2	42,387.85	42,291.63	42,278.45	42,289.09	42,339.90	42,338.01	42,300.76
C2_10_9	16,421.32	16,370.44	16,421.14	16,481.81	16,372.32	16,372.32	16,432.53
R1_10_1	53,687.82	53,560.85	53,772.62	53,846.08	53,676.62	53,615.67	53,657.99
R2_10_1	42,188.86	42,188.86	42,261.34	42,259.70	42,196.47	42,188.86	42,219.21
R2_10_2	33,634.05	33,554.71	33,570.65	33,912.29	33,550.62	33,512.83	33,567.91
R2_10_3	24,965.63	24,954.81	25,029.43	25,106.05	24,989.17	24,940.32	25,053.80
R2_10_4	17,926.45	17,997.69	19,180.23	19,421.25	18,262.09	17,959.91	18,039.77
R2_10_5	36,243.51	36,251.43	36,289.00	36,352.12	36,250.95	36,232.18	36,335.72
R2_10_6	30,123.53	30,099.45	30,270.10	30,309.43	30,102.91	30,091.93	30,223.14
R2_10_7	23,257.36	23,344.99	23,730.70	23,636.36	23,270.90	23,355.92	23,381.36
R2_10_8	17,585.79	17,495.51	18,018.54	17,859.14	17,526.72	17,524.42	17,598.63
R2_10_9	33,064.26	33,019.95	33,146.76	33,116.32	33,025.59	33,002.36	33,131.99
R2_10_10	30,257.93	30,242.35	30,428.55	30,499.52	30,270.36	30,215.24	30,598.69
RC2_10_3	20,050.71	20,084.70	20,132.61	20,229.46	20,068.74	20,076.27	20,053.78
RC2_10_6	26,910.35	26,864.81	27,097.58	27,086.98	26,883.75	26,797.76	26,965.51
RC2_10_7	25,116.15	25,112.77	25,380.63	25,471.85	25,162.18	25,128.19	25,295.67
RC2_10_8	23,845.66	23,709.29	24,013.56	24,063.00	23,822.03	23,796.20	23,787.26
RC2_10_9	23,112.13	23,063.76	23,292.42	23,297.83	23,089.01	23,061.90	23,116.15
RC2_10_10	22,048.98	21,965.94	22,225.09	22,347.35	22,004.78	21,999.00	22,076.90

The best result is rendered in boldface; WB indicates the world's best T beaten by PMA-VRPTW

**Fig. 5** The convergence times (in seconds) of PMA-VRPTW with various co-operation schemes applied, averaged for each GH subclass: **a** $\delta = 2$, **b** $\delta = 20$ (Color figure online)

this scheme significantly affects the algorithm convergence capabilities. In contrary, KS scheme distributes the best individuals between islands relatively fast, and guides the search efficiently towards the most-promising parts of the search space.

Since the execution time of the algorithm (and more importantly—time of converging to acceptable solutions) is an important criterion in case of real-life applications,

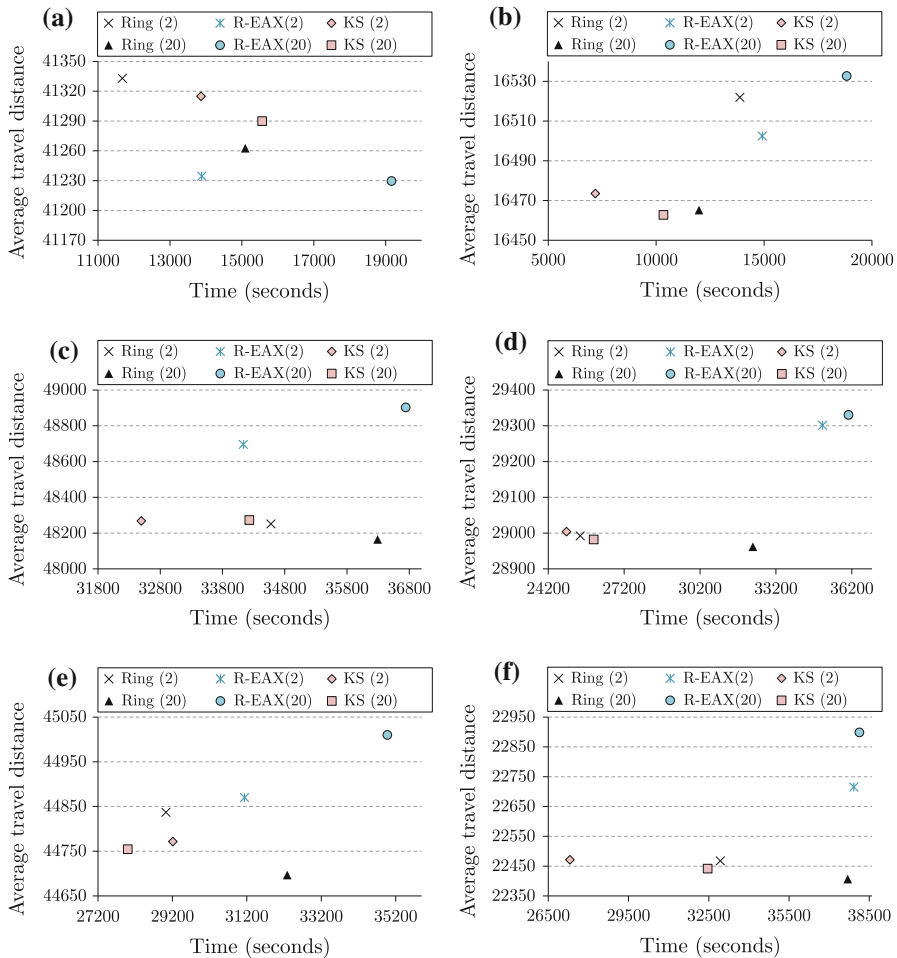


Fig. 6 The front (the average T vs. the average PMA-VRPTW convergence time in seconds) for each GH subclass (the closer to the point (0, 0), the better): **a** C1, **b** C2, **c** R1, **d** R2, **e** RC1, and **f** RC2 (δ is given in brackets)

determining a proper co-operation scheme becomes a critical issue for emerging PEAs. In Fig. 6, we visualize a trade-off between minimizing the convergence time of PMA-VRPTW, and obtaining solutions with the highest possible quality. Clearly, R-EAX is the most time-consuming scheme (and introduces the largest co-operation overhead) among the considered ones, and choosing this scheme is advantageous only for C1 tests. It is easy to see that applying KS (for both $\delta = 2$ and $\delta = 20$) results in the fastest convergence of PMA-VRPTW in most GH subclasses. There are however subclasses for which Ring (with $\delta = 20$) gave higher-quality asymptotic solutions on average (R1, RC1, RC2), but the convergence time is significantly larger here compared with the KS scheme (see Fig. 6c–f).

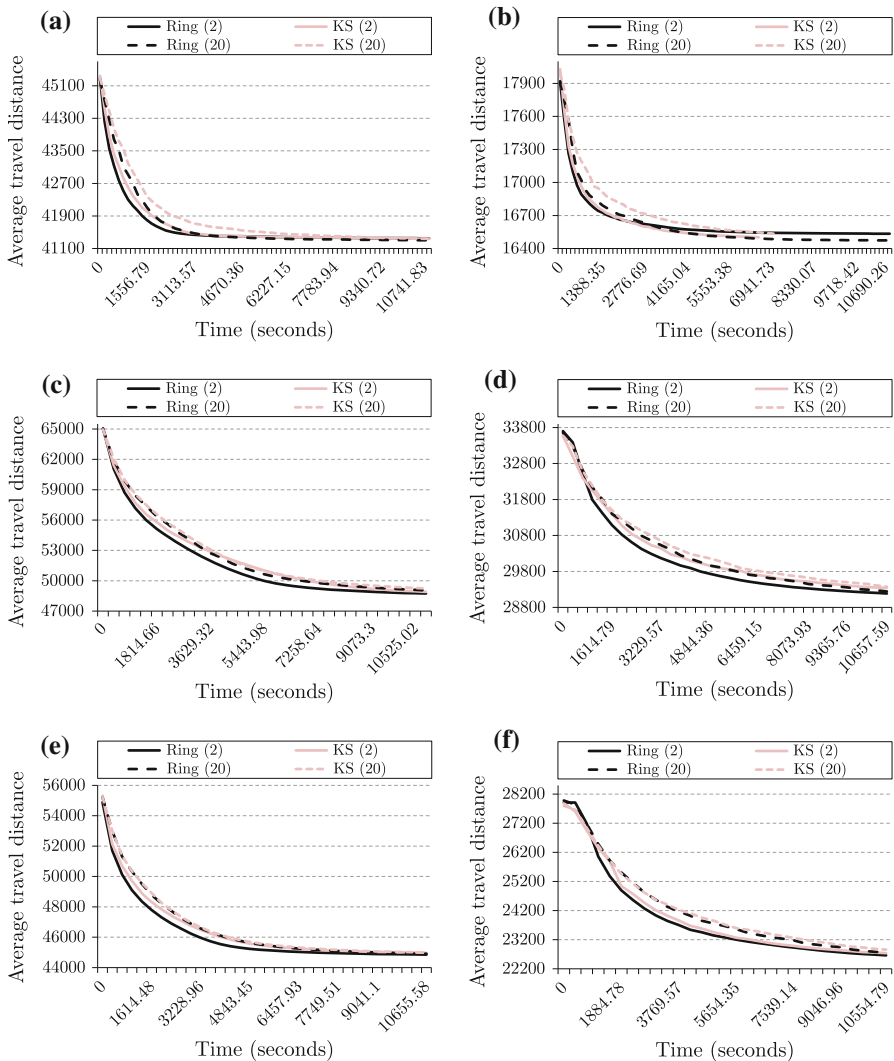


Fig. 7 The travel distance averaged for each GH subclass obtained using PMA-VRPTW (with the best co-operation schemes applied) within the initial 180 min. of execution: **a** C1, **b** C2, **c** R1, **d** R2, **e** RC1, and **f** RC2 (δ is given in brackets)

In Fig. 7, we present the average travel distance (for best co-operation schemes) for each subclass during the initial 180 min. of the PMA-VRPTW execution. This complements the results shown in Fig. 6, and gives a better insight into the PMA-VRPTW convergence capabilities. The small migration intervals are very beneficial at the beginning of the optimization process (see Fig. 7). If islands co-operate frequently, then the gained knowledge is distributed very fast. It can be later utilized to exploit known (already optimized) solutions. A noteworthy feature of the co-operation schemes with a larger migration interval applied (KS and Ring with $\delta = 20$) is their

Table 10 The guidelines for choosing the co-operation scheme in PMA-VRPTW

GH subclass ↓	Minimize τ_P ?			
	Yes		No	
	Scheme	δ	Scheme	δ
C1	R-EAX	2	R-EAX	20
C2	KS	2	KS	20
R1	KS	2	Ring	20
R2	KS	20	Ring	20
RC1	KS	20	Ring	20
RC2	KS	2	Ring	20

ability to balance exploration and exploitation of the search space. Thus, they are able to outperform the other schemes asymptotically, and to give the best solutions in a longer optimization horizon.

6 Conclusions and Future Work

In this paper, we investigated the impact of the applied co-operation scheme (migration topology, interval, handling of immigrants, and emigrants) on the search capabilities of our parallel MA to solve the VRPTW. The extensive experimental study which comprised more than 1,584,000 CPU hours on an SMP cluster gave a detailed insight into the schemes' characteristics and capabilities. Since real-life logistic and distribution problems are diverse at their heart (they differ by the customer locations, characteristics of time windows, vehicle capacities, and many more), we solved and analyzed the most-demanding 1000-customer benchmark tests reflecting various scheduling circumstances.

Concluding from our experiments, we provide the guidelines for choosing a proper migration topology and interval (see Table 10). If the execution (convergence) time of PMA-VRPTW is to be as minimum as possible, then the frequent co-operation should be favored to guide the search efficiently (KS). If there is no strict upper bound for the running time of PMA-VRPTW, then applying the rarer co-operation intervals benefits from a broader exploration of the search space (Ring), and gives the best asymptotic results.

Our ongoing works are focused on combining two most successful co-operation schemes—KS and Ring—into an adaptive scheme. This scheme will dynamically control itself (its migration topology and interval) according to the current state of exploring the search space. Also, we work on the adaptive MA, in which the parameters (including the population size, selection scheme, and more) are controlled (and appropriately changed) on the fly. This approach will mitigate the necessity of performing a time-consuming tuning process of PMA-VRPTW parameters. Finally, we plan to apply PMA-VRPTW for other VRPs—especially the pickup and delivery problem with time windows.

Acknowledgments This research was performed using the infrastructure supported by POIG.02.03.01-24-099/13 Grant: “GeCONiL—Upper Silesian Center for Computational Science and Engineering”. Also,

we thank the following computing centers where the computations of our Project were carried out: Academic Computer Centre in Gdańsk TASK, Academic Computer Centre CYFRONET AGH, Kraków, Interdisciplinary Centre for Mathematical and Computational Modeling, Warsaw University, Wrocław Centre for Networking and Supercomputing. This research was supported by the National Science Centre under research Grant No. DEC-2013/09/N/ST6/03461.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

1. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. *Trans. Evol. Comp.* **6**(5), 443–462 (2002)
2. Alba, E., Troya, J.M.: A survey of parallel distributed genetic algorithms. *Complexity* **4**(4), 31–52 (1999)
3. Baldacci, R., Mingozzi, A., Roberti, R.: New route relaxation and pricing strategies for the vehicle routing problem. *Oper. Res.* **59**(5), 1269–1283 (2011)
4. Baldacci, R., Mingozzi, A., Roberti, R.: Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *Eur. J. Oper. Res.* **218**(1), 1–6 (2012)
5. Banos, R., Ortega, J., Gil, C., Márquez, A.L., de Toro, F.: A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. *Comput. Ind. Eng.* **65**(2), 286–296 (2013)
6. Bard, J.F., Kontoravdis, G., Yu, G.: A branch-and-cut procedure for the vehicle routing problem with time windows. *Transp. Sci.* **36**(2), 250–269 (2002)
7. Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega* **34**(3), 209–219 (2006)
8. Blocho, M.: A Parallel Memetic Algorithm for Solving the Vehicle Routing Problem with Time Windows. Ph.D. thesis, Silesian University of Technology, Gliwice, Poland (2013)
9. Blocho, M., Czech, Z.: A parallel algorithm for minimizing the number of routes in the vehicle routing problem with time windows. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waniowski, J. (eds.) *Parallel Processing and Applied Mathematics. Lecture Notes in Computer Science*, vol. 7203, pp. 255–265. Springer, Berlin (2012)
10. Blocho, M., Czech, Z.: A parallel EAX-based algorithm for minimizing the number of routes in the vehicle routing problem with time windows. In: 2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICES), pp. 1239–1246 (2012)
11. Blocho, M., Czech, Z.J.: An improved route minimization algorithm for the vehicle routing problem with time windows. *Studia Informatica* **32**(99), 5–19 (2010)
12. Blocho, M., Czech, Z.J.: A parallel memetic algorithm for the vehicle routing problem with time windows. In: *Proceedings of the 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC '13)*, pp. 144–151 (2013)
13. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, part II: metaheuristics. *Transp. Sci.* **39**(1), 119–139 (2005)
14. Cantu-Paz, E.: A survey of parallel genetic algorithms. *Calcul. Paralleles* **10**, 141–171 (1998)
15. Chabrier, A.: Vehicle routing problem with elementary shortest path based column generation. *Comput. Oper. Res.* **33**(10), 2972–2990 (2006). Part Special Issue: Constraint Programming
16. Chiang, W.C., Russell, R.: Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Ann. Oper. Res.* **63**(1), 3–27 (1996)
17. Cipolla, M., Bosco, G.L., Millonzi, F., Valenti, C.: An island strategy for memetic discrete tomography reconstruction. *Inf. Sci.* **257**, 357–368 (2014)
18. Coltori, D., Rizzoli, A.E.: Ant colony optimization for real-world vehicle routing problems. *SIGEVOLUTION* **2**(2), 2–9 (2007)
19. Cordeau, J.F., Desautels, G., Desrosiers, J., Solomon, M.M., Soumis, F.: The vehicle routing problem. In: *Chap. VRP with Time Windows*, pp. 157–193. Society for Industrial and Applied Mathematics (2001)
20. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. *Manag. Sci.* **6**(1), 80–91 (1959)

21. Desrochers, M., Desrosiers, J., Solomon, M.: A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.* **40**(2), 342–354 (1992)
22. El-Sherbeny, N.A.: Vehicle routing with time windows: an overview of exact, heuristic and metaheuristic methods. *J. King Saud Univ. Sci.* **22**(3), 123–131 (2010)
23. Feillet, D., Dejax, P., Gendreau, M., Gueguen, C.: An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks* **44**(3), 216–229 (2004)
24. Gehring, H., Homberger, J.: A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: *Proceedings of EUROGEN99-Short Course on Evolutionary Algorithms in Engineering and Computer Science*, pp. 57–64 (1999)
25. Ghoseiri, K., Ghannadpour, S.F.: Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Appl. Soft Comput.* **10**(4), 1096–1107 (2010)
26. Gomez, C., Cruz-Reyes, L., González, J.J., Fraire, H.J., Pazos, R.A., Martinez, J.J.: Ant colony system with characterization-based heuristics for a bottled-products distribution logistics system. *J. Comput. Appl. Math.* **259**(Part B(0)), 965–977 (2014)
27. Guan, X., Zhang, X., Han, D., Zhu, Y., Lv, J., Su, J.: A strategic flight conflict avoidance approach based on a memetic algorithm. *Chin. J. Aeronaut.* **27**(1), 93–101 (2014)
28. Ho, S., Haugland, D.: A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Comput. Oper. Res.* **31**(12), 1947–1964 (2004)
29. Hosny, M.I., Mumford, C.L.: The single vehicle pickup and delivery problem with time windows: intelligent operators for heuristic and metaheuristic algorithms. *J. Heuristics* **16**(3), 417–439 (2010)
30. Hu, W., Liang, H., Peng, C., Du, B., Hu, Q.: A hybrid chaos-particle swarm optimization algorithm for the vehicle routing problem with time window. *Entropy* **15**(4), 1247–1270 (2013)
31. Irnich, S., Villeneuve, D.: The shortest-path problem with resource constraints and k -cycle elimination for $k \leq 3$. *INFORMS J. Comput.* **18**(3), 391–406 (2006)
32. Jin, Y., Hao, J.K., Hamiez, J.P.: A memetic algorithm for the minimum sum coloring problem. *Comput. Oper. Res.* **43**, 318–327 (2014)
33. Kallehauge, B.: Formulations and exact algorithms for the vehicle routing problem with time windows. *Comput. Oper. Res.* **35**(7), 2307–2330 (2008)
34. Kawulok, M., Nalepa, J.: Support vector machines training data selection using a genetic algorithm. In: Gimelfarb, G., Hancock, E., Imiya, A., Kuijper, A., Kudo, M., Omachi, S., Windeatt, T., Yamada, K. (eds.) *Structural, Syntactic, and Statistical Pattern Recognition. Lecture Notes in Computer Science*, vol. 7626, pp. 557–565. Springer, Berlin (2012)
35. Kindervater, G., Savelsbergh, M.: Vehicle routing: handling edge exchanges. In: Aarts, E., Lenstra, J. (eds.) *Local Search in Combinatorial Optimization*, pp. 337–360. Wiley, New York (1997)
36. Kolen, A.W.J., Kan, A.H.G.R., Trienekens, H.W.J.M.: Vehicle routing with time windows. *Oper. Res.* **35**(2), 266–273 (1987)
37. Larsen, J.: Refinements of the column generation process for the vehicle routing problem with time windows. *J. Syst. Sci. Syst. Eng.* **13**(3), 326–341 (2004)
38. Li, Y., Jiao, L., Li, P., Wu, B.: A hybrid memetic algorithm for global optimization. *Neurocomputing* **134**, 132–139 (2014)
39. Li, Y., Li, P., Wu, B., Jiao, L., Shang, R.: Kernel clustering using a hybrid memetic algorithm. *Nat. Comput.* **12**(4), 605–615 (2013)
40. Liu, Y.Y., Wang, S.: A scalable parallel genetic algorithm for the generalized assignment problem. *Parallel Comput.* (in press) (2014). doi:[10.1016/j.parco.2014.04.008](https://doi.org/10.1016/j.parco.2014.04.008)
41. Marinaki, M., Marinakis, Y.: An island memetic differential evolution algorithm for the feature selection problem. In: *Proceedings of the NICSO, SCI*, vol. 512, pp. 29–42. Springer, Berlin (2014)
42. Mirsoleimani, S.A., Karami, A., Khunjush, F.: A parallel memetic algorithm on GPU to solve the task scheduling problem in heterogeneous environments. In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*, pp. 1181–1188. ACM (2013)
43. Nagata, Y.: New EAX crossover for large TSP instances. In: Runarsson, T., Beyer, H.G., Burke, E., Merelo-Guervs, J., Whitley, L., Yao, X. (eds.) *Parallel Problem Solving from Nature—PPSN IX. Lecture Notes in Computer Science*, vol. 4193, pp. 372–381. Springer, Berlin (2006)
44. Nagata, Y.: Edge assembly crossover for the capacitated vehicle routing problem. In: Cotta, C., Hemert, J. (eds.) *Evolutionary Computation in Combinatorial Optimization. Lecture Notes in Computer Science*, vol. 4446, pp. 142–153. Springer, Berlin (2007)

45. Nagata, Y., Bräysy, O.: A powerful route minimization heuristic for the vehicle routing problem with time windows. *Oper. Res. Lett.* **37**(5), 333–338 (2009)
46. Nagata, Y., Bräysy, O., Dullaert, W.: A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Comput. Oper. Res.* **37**(4), 724–737 (2010)
47. Nalepa, J., Blocho, M., Czech, Z.: Co-operation schemes for the parallel memetic algorithm. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waniowski, J. (eds.) *Parallel Processing and Applied Mathematics. Lecture Notes in Computer Science*, pp. 191–201. Springer, Berlin (2014)
48. Nalepa, J., Czech, Z.J.: Adaptive threads co-operation schemes in a parallel heuristic algorithm for the vehicle routing problem with time windows. *Theor. Appl. Inform.* **24**(3), 191–203 (2012)
49. Nalepa, J., Czech, Z.J.: A parallel heuristic algorithm to solve the vehicle routing problem with time windows. *Studia Informatica* **33**(1), 91–106 (2012)
50. Nalepa, J., Czech, Z.J.: New selection schemes in a memetic algorithm for the vehicle routing problem with time windows. In: Tomassini, M., Antonioni, A., Daolio, F., Buesser, P. (eds.) *Adaptive and Natural Computing Algorithms. Lecture Notes in Computer Science*, vol. 7824, pp. 396–405. Springer, Berlin (2013)
51. Nalepa, J., Kawulok, M.: A memetic algorithm to select training data for support vector machines. In: *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation (GECCO '14)*, pp. 573–580. ACM, New York, NY, USA (2014)
52. Oh, S.K., Kim, W.D., Pedrycz, W., Seo, K.: Fuzzy radial basis function neural networks with information granulation and its parallel genetic optimization. *Fuzzy Sets Syst.* **237**, 96–117 (2014)
53. Pang, K.W.: An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints. *Expert Syst. Appl.* **38**(9), 11,939–11,946 (2011)
54. Petch, R., Salhi, S.: A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discret. Appl. Math.* **133**(13), 69–92 (2003)
55. Potvin, J.Y., Rousseau, J.M.: A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *Eur. J. Oper. Res.* **66**(3), 331–340 (1993)
56. Potvin, J.Y., Rousseau, J.M.: An exchange heuristic for routeing problems with time windows. *J. Oper. Res. Soc.* **46**(12), 1433–1446 (1995)
57. Repoussis, P., Tarantilis, C., Ioannou, G.: Arc-guided evolutionary algorithm for the vehicle routing problem with time windows. *IEEE Trans. Evol. Comput.* **13**(3), 624–647 (2009)
58. Righini, G., Salani, M.: Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discret. Optim.* **3**(3), 255–273 (2006). *Graphs and Combinatorial Optimization The Cologne/Twente Workshop on Graphs and Combinatorial Optimization*
59. Roberge, V., Tarbouchi, M., Okou, F.: Strategies to accelerate harmonic minimization in multilevel inverters using a parallel genetic algorithm on graphical processing unit. *IEEE Trans. Power Electron.* **29**(10), 5087–5090 (2014)
60. Sarkar, B.K., Sana, S.S., Chaudhuri, K.: Selecting informative rules with parallel genetic algorithm in classification problem. *Appl. Math. Comput.* **218**(7), 3247–3264 (2011)
61. Segredo, E., Segura, C., Leon, C.: A multiobjectivised memetic algorithm for the frequency assignment problem. In: *2011 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1132–1139 (2011)
62. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **35**(2), 254–265 (1987)
63. Sudholt, D.: Parallel evolutionary algorithms. In: Kacprzyk, J., Pedrycz, W. (Eds.) *Handbook of Computational Intelligence*. Springer, Netherlands (2014) (in press)
64. Tavares, L., Lopes, H., Lima, C.: Construction and improvement heuristics applied to the capacitated vehicle routing problem. In: *World Congress on Nature Biologically Inspired Computing, 2009 (NaBIC 2009)*, pp. 690–695 (2009)
65. Tripathy, P., Dash, R., Tripathy, C.: A genetic algorithm based approach for topological optimization of interconnection networks. *Procedia Technol.* **6**, 196–205 (2012)
66. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Comput. Oper. Res.* **40**(1), 475–489 (2013)
67. Xhafa, F., Duran, B.: Parallel memetic algorithms for independent job scheduling in computational grids. In: Cotta, C., van Hemert, J. (eds.) *Recent Advances in Evolutionary Computation for Combinatorial Optimization, Studies in Computational Intelligence*, vol. 153, pp. 219–239. Springer, Berlin (2008)

68. Yu, B., Yang, Z., Sun, X., Yao, B., Zeng, Q., Jeppesen, E.: Parallel genetic algorithm in bus route headway optimization. *Appl. Soft Comput.* **11**(8), 5081–5091 (2011)
69. Zhong, Y., Pan, X.: A hybrid optimization solution to VRPTW based on simulated annealing. In: 2007 IEEE International Conference on Automation and Logistics, pp. 3113–3117 (2007)